LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORTIMA
“Penyelesaian *Crossword Puzzle* dengan Algoritma *Brute Force*”

disusun oleh:

Erma Safira Nurmasyita

13516072

K03 Teknik Informatika 2016

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2018

Penyelesaian *Crossword Puzzle* dengan Algoritma *Brute Force*

Papan *Crossword Puzzle* disimpan dalam matriks berisi *char*. Semua *grid* kosong dienumerasi pada *list of Grid*, sebuah tipe bentukan dengan properti *tuple rowcol*, *orientation*, *length*, dan *previous grid blocks*. Program diimplementasikan dengan bahasa Python.

Algoritma *Brute Force*

1. List *grid* kosong diurutkan membesar sesuai nilai *tuple row* dan *column grid* pada papan *Puzzle*.
2. Mencoba semua kata pada sebuah *grid*. Pada proses pertama dimulai dari *grid* pada urutan pertama.
3. Jika ditemukan kata yang memenuhi sebuah *grid*, kata tersebut dituliskan ke dalam *grid* pada papan *Puzzle*.
4. Mencobakan semua kata yang belum tertulis pada papan pada *grid* selanjutnya.
5. Jika tidak terdapat kata yang memenuhi sebuah *grid*, kembali ke *grid* sebelumnya dan menghapus kata yang tertulis pada *grid* tersebut (tanpa menghapus huruf yang bersimpangan dengan kata pada *grid* lain). Kemudian mencoba *grid* dengan kata lainnya yang memenuhi.
6. Proses diulang hingga pengisian *grid* terakhir. Jika *grid* terakhir telah terpenuhi, *Puzzle* terselesaikan.

Source Code Program

```
'''
    TUGAS KECIL 1 STRATEGI ALGORITMA
    CROSSWORD PUZZLE SOLVER
    ERMA SAFIRA NURMASYITA
    13516072
    23 JANUARI 2018
'''
import sys
import operator
import time

### Data Structure ###
class Grid(object):
# Stores the properties of empty grids in the Puzzle
    def __init__(self, rowcol = (0, 0), length = 0, orient = None, prev = []):
        self.rowcol = rowcol # (row, column)
        self.length = length # empty grid length
        self.orient = orient # vertical or horizontal orientation
        self.prev = prev     # previous grid blocks before word placement

### Primitives ###
## Initialization ##
def FileToPuzzle(filename):
```

```
# Load puzzle from file
    global Puzzle, N, WordList
    file = open(filename, "r")
    filelist = file.read().splitlines()
    file.close()
    # number of board row and column
    N = int(filelist[0])
    # form puzzle board
    for i in range(1, N+1):
        column = list(filelist[i])
        Puzzle.append(column)
    # list of words
    answer = filelist[N+1]
    WordList = answer.split(';')

def enumGrid(Puzzle, N):
# Enumerate the empty grids into GridList
    GridList = []
    j = 0
    gridlength = 0
    # Detecting blank grids horizontally
    for i in range(0 , N):
        while (j < N):
            if Puzzle[i][j] == '-':
                while True:
                    gridlength += 1
                    j += 1
                    if (j >= N or Puzzle[i][j] == '#'):
                        if gridlength > 1:
                            rowcol = (i, j-gridlength)
                            GridList.append(Grid(rowcol, gridlength, 'hor', []))
                        break
                gridlength = 0
            else:
                j += 1
        j = 0
    # Detecting blank grids vertically
    i = 0
    gridlength = 0
    for j in range(0 , N):
        while (i < N):
            if Puzzle[i][j] == '-':
                while True:
                    gridlength += 1
                    i += 1
                    if (i >= N or Puzzle[i][j] == '#'):
                        if gridlength > 1:
                            rowcol = (i-gridlength, j)
                            GridList.append(Grid(rowcol, gridlength, 'ver', []))
                        break
                gridlength = 0
            else:
                i += 1
        i = 0
    # sort grids based on row & column tuple
    GridList.sort(key=operator.attrgetter('rowcol'))
    return GridList

def initPuzzle(filename):
# Load puzzle from file and enumerate grids which can be filled out
    global Puzzle, N, GridList, WordList, UsedWordList
    Puzzle = []
    WordList = []
    FileToPuzzle(filename)
    GridList = enumGrid(Puzzle, N)
    UsedWordList = [False]*len(WordList) # Save each word's usage status

## Crossword Solver ##
def GridToWord(grid):
# Transform Grid in Puzzle board into list of char
    row = grid.rowcol[0]
    col = grid.rowcol[1]
    Word = []
    if grid.orient == 'hor':
        for i in range (0, grid.length):
            Word.append(Puzzle[row][col+i])
    else:
        for i in range (0, grid.length):
            Word.append(Puzzle[row+i][col])
    return Word

def isWordFit(Grid, Word):
# Check whether a Word fits to an unsolved grid
# Precond: len(Word) and Grid.length is not zero
    if Grid.length == len(Word):
        i = 0
        isFit = True
        GridWord = GridToWord(Grid)
        while isFit and i < Grid.length:
            if GridWord[i] != '-' and GridWord[i] != Word[i]:
                isFit = False
            else:
                i += 1
        return isFit
    else:
        return False

def WriteWordinGrid(Word, Grid):
# Write a Word to the Grid in puzzle board
# Precond: Word is fit for Grid
    global Puzzle
    row = Grid.rowcol[0]
    col = Grid.rowcol[1]
    j = 0
    (Grid.prev) = []
    if (Grid.orient == 'hor'):
        for i in range(0, Grid.length):
            (Grid.prev).append(Puzzle[row][col+i])
            Puzzle[row][col+i] = Word[j]
            j += 1
    else:
        for i in range(0, Grid.length):
            (Grid.prev).append(Puzzle[row+i][col])
            Puzzle[row+i][col] = Word[j]
            j += 1

def PuzzleSolver(idxGrid):
# Solve Puzzle with brute force algorithm with recursive approach
    global GridList, UsedWordList, issolved, stop_time
    if (idxGrid == len(GridList)):  # Basis: All grids are solved
        stop_time = time.time()
        issolved = True
        printPuzzle(Puzzle)
    else:
        for idxWord in range(0, len(WordList)):
            Grid, Word = GridList[idxGrid], WordList[idxWord]
            if (not UsedWordList[idxWord] and isWordFit(Grid, Word)):
                # Write Word to Grid with index idxGrid
                WriteWordinGrid(Word, Grid)
                UsedWordList[idxWord] = True
                if not issolved:    # Recurrence
                    PuzzleSolver(idxGrid + 1)
                # Clear Word from Grid with index idxGrid
                WriteWordinGrid(Grid.prev, Grid)
                UsedWordList[idxWord] = False
```

```
def solvePuzzle():
# Procedure for puzzle solver; print and timer
    global issolved, time
    issolved = False    # Puzzle board solved-status
    start_time = time.time()
    PuzzleSolver(0)     # Algorithm to solve
    time = stop_time - start_time

## Output ##
def printPuzzle(PuzzleBoard):
# Print Crossword Puzzle
    i, j = 0, 0
    for i in range(0, N):
        for j in range(0, N):
            print("%c " % PuzzleBoard[i][j], end="")
        print()

### Main Program ###
initPuzzle(sys.argv[1])   # sys.argv[1] = <filename.txt>
print("-- CROSSWORD PUZZLE SOLVER --")
solvePuzzle()   # Solve, print, and count time execution for solving puzzle
print("*** Crossword Puzzle has been solved for %s sec(s) ***" % (time))
```

Input **dan** ***Output*** **Program**

Data test 1

```
> python crosswordsolver.py puzzle1.txt

-- CROSSWORD PUZZLE SOLVER --

# A N T S # S O U N D # A G E N T

D # # R # # O # # # E # # U # # R

E # # A # F I S H # N O R T H # O

A G E N T # L # # # G # # S # # P

T # # S # E # # # # U # B # B T I

H O R M O N O M I M E T I C # # C

# D # I # E # # # A # # O # C # S

H O S T # M # P # R # A L G A E #

# R # # # I # A # I # # O # T # #

R # # F E E D I N G # E G G S # G

A # # A # S # N # O # # I # # # E
```

```
S O U T H # W # A L T O C I D # N
H # # A # # A # # D # # A # O # E
# # P L A N T # # S U N L I G H T
P # # # N # E # # # # # # # # # I
E # # # T I R E S # # # # # # # C
T E X A S # # # # # V E C T O R S
*** Crossword Puzzle has been solved for 0.008056402206420898 sec(s) ***
```

Data test 2

```
> python crosswordsolver.py puzzle2.txt
-- CROSSWORD PUZZLE SOLVER --
D E A N # N A B S # K H A K I
E L M O # E L S A # V A L I D
P L O T # S C A R # E M O T E
T A K E S T O # A R T L E S S
# # # P I L A F # I C E # # #
O C T A N E # L I G H T S U P
P R I D E # Y O R E # S E L L
T U T # W E A R I L Y # R T E
I S L A # K R I S # U H U R A
C H E C K I N S # A R O M A S
# # # R A N # T U F T S # # #
R A L E I G H # S I S T E R S
A R E A S # A G U E # E T A S
T I N G E # N E A L # S N I T
A D D E R # G O L D # S A N S
*** Crossword Puzzle has been solved for 29.932878255844116 sec(s) ***
```

Data test 3

```
> python crosswordsolver.py puzzle3.txt
-- CROSSWORD PUZZLE SOLVER --
# F O R E S T # # S E D U C E
# L # H # H # G # T # E # H #
S U R E F I R E # R E C O I L
# O # N # V # R # A # O # C #
```

```
F R A I S E # M A N D R A K E
# E # U # # # A # G # U # E #
# S E M I F I N A L # M I N T
# C # # # O # # # E # # # C #
P E E R # R E F U R B I S H #
# N # E # B # E # # # N # A #
S T E A D I E R # M I S U S E
# T # D # D # U # E # P # S #
B U R I E D # L A T H E R E D
# B # E # E # E # H # C # U #
S E A S O N # # P O S T E R #
*** Crossword Puzzle has been solved for 0.004010200500488281 sec(s) ***
```

Data test 4

```
> python crosswordsolver.py puzzle4.txt
-- CROSSWORD PUZZLE SOLVER --
# # # B R I D A L # T R A P #
Y # D # E # O # Y # W # # I #
U B E N D # D I S T E M P E R
M # L # R # D # O # E # A # O
C L I M A T E # L I T U R G Y
H # B # W # R # # # E # A # A
A C E S # H I S T O R I C A L
# # R # D # N # H # S # H # #
F L A M I N G O E S # S U V A
O # T # V # # # O # A # T # R
R A I S I N S # R A C K I N G
T # O # S # W # I # U # S # Y
H A N D I C A P S # M O T E L
# R # # V # M # T # E # S # E
# C A K E # P O S I N G # # #
*** Crossword Puzzle has been solved for 0.0020046234130859375 sec(s) ***
```

Data test 5

```
> python crosswordsolver.py puzzle5.txt
```

```
-- CROSSWORD PUZZLE SOLVER --
M I L E # # F L I C # R E I N
O D O R S # E A C H # O R C A
M E N S A # A N K A # G R I M
S A G # L O R D # S T E E L E
# # # A I D E # S T U R D Y #
S H E L V E D # E I N # # # #
H O N D A # # G E T G O I N G
O B O E # B E A D Y # T O O N
D O W N W A R D # # U H U R A
# # # # R C A # F O R E S A W
# F R E A K S # L E A R # # #
P L U M P S # T O R N # Q U E
E Y R E # O P E C # U S U R P
S E A R # F I N K # S T I L E
O R L Y # F E D S # # A P S E
*** Crossword Puzzle has been solved for 33.8730103969574 sec(s) ***
```

Tabel Penilaian

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil *running*	√	
3. Program dapat membaca *file* masukan dan menuliskan luaran	√	
4. Solusi *crossword puzzle* benar untuk semua data test	√	